# 基于OpenCL的射电干涉阵成像gridding算法实现*

冯勇[1]，王锋[1,2,3]，邓辉[1,2]，卫守林[1]，梅盈[1,2,3]，戴伟[1,3] 石聪明[1]
(1.昆明理工大学 云南省计算机技术应用重点实验室，云南 昆明 650500
2.广州大学 天体物理中心/物理与电子工程学院，广东 广州 510006
3.中国科学院云南天文台 云南 昆明 650011)

**摘 要：**天文软件开发与应用中迫切需要在单机环境下进行高性能的科学数据处理工作，由于机器配置不同，采用传统的CUDA+GPU技术存在明显的局限，不利于天文软件的快速移植和无缝运行。针对明安图频谱射电日像仪（MingantU SpEctral Radioheliograph，MUSER）数据处理中的网格化（Gridding）算法，采用并行计算OpenCL技术进行多线程编程实现。实验结果表明，基于OpenCL实现的gridding算法不仅能够在多种GPU上运行，而且能够在纯CPU上运行。当选择在GPU上执行时，算法的执行效率与基于CUDA实现的gridding算法执行效率大致相当，但算法不局限于NVIDIA GPU，解决了算法对CUDA+GPU的依赖；同时算法也能在纯CPU上较快速地执行，适用于单机模式下进行天文软件的开发和测试，也便于天文软件的应用与推广。

**关键词：**Gridding；并行计算； OpenCL；MUSER
**中图分类号：**TP311.1　**文献标识码：**A　**文章编号：**1007-2276-(2004)4-0338-05

## 0 引 言　[1]一级标题：小四，黑体，上下空一行

中国新一代厘米-分米波射电日像仪—明安图频谱射电日像仪(MingantU SpEctral Radioheliograph, MUSER)是基于~~利用~~综合孔径成像原理实现在频率范围 0.4 - 15.0 GHz 内对太阳进行高分辨率（时间、空间）观测并成像的射电望远镜[1,2]。

MUSER 主要由天线(天线阵)、接收系统、数据处理系统三部分组成。天线阵由低频阵(MUSER-I，共 40 面天线)和高频阵(MUSER-II，共 60 面天线)两部分组成，天线阵的拓扑结构为螺旋阵[3,4]。在 MUSER 观测过程中，低频阵和高频阵分别每 3 ms 产生一个数据帧，每个数据帧分别用 100 KBytes 和 256 KBytes 存储，数据流量分别为 32 MB/s 和 86 MB/s，以每天观测 10 小时为例，每日原始观测数据量分别约为 1.2 TB 和 3.2 TB[5,6]。

如此海量的天文数据，对于 MUSER 数据处理工作而言是一个巨大的挑战，但挑战与机遇并存。在前期研究工作中，为了满足数据处理需求，研究人员一方面采用分布式计算技术，设计并实现了一套分布式数据处理执行框架 OpenCluster[7]；另一方面采用并行计算技术并借助高性能计算设备，专注于底层算法研究，实现了一套高效的射电干涉阵数据处理管线[8]。特别地，采用 MPI（Message Passing Interface，消息传递接口）、CUDA（Compute Unified Device Architecture，统一计算设备架构）以及 OpenCL（Open Computing Language，开放运算语言）技术已经在天文中有相关的应用，并已经取得一定的成果[9,10,11]。证明这些技术稳定性与性能可以满足天文应用软件开发的要求。

为了便于 MUSER 数据处理系统的开发、部署、应用以及推广，同时满足单机环境下仍旧能够进行高性能的成像数据处理工作。在前期对 OpenCL 技术研究基础上，进一步采用 OpenCL 技术对射电干涉阵成像中的网格化（Gridding）算法进行并行优化，在保证 gridding 算法执行效率的同时，提升算法对各种硬件平台的适应性，便于后期 MUSER 数据处理软件的应用与推广。

---

*基金项目：国家重点研发计划(2016YFE0100300)资助，国家自然科学基金委员会-中国科学院天文联合基金项目(No.U1531132,U1631129) 资助，国家自然科学基金项目(No.11403009,11463003,11773012) 资助，赛尔网络下一代互联网技术创新项目（No.NGII20170204）资助.
**收稿日期：**2018-03-06；　修订日期：2018-03-06
**作者简介：**冯勇，男，硕士，研究方向：计算机应用技术. Email:fengyong@cnlab.net
**通讯作者：**王锋，男，教授，研究方向：天文技术与方法. Email:wangfeng@cnlab.net

# 1 基于OpenCL的gridding算法实现

## 1.1 Gridding 算法

射电干涉阵成像过程是基于综合孔径成像原理对采样的可见度数据进行网格化(Gridding)、快速傅里叶变换(Fast Fourier Transformation，FFT)、去卷积(Deconvolution)等操作，最后产生射电源的观测图像[12]。

为了借助离散傅里叶变换（Discrete Fourier Transform，DFT）的快速方法—FFT—的计算效率，进行快速成像，射电干涉阵采集到的可见度数据必须落在一个均匀划分的网格上（如图 1 左图所示）。实际中，由于 UV 采样点分布不均匀，干涉仪得到的是非均匀采样可见度数据（Non-uniform Sampled Visibility Data）(如图 1 右图所示)，将非均匀采样可见度数据插值到均匀划分的网格点上的过程称为网格化（Gridding）[13,14]。

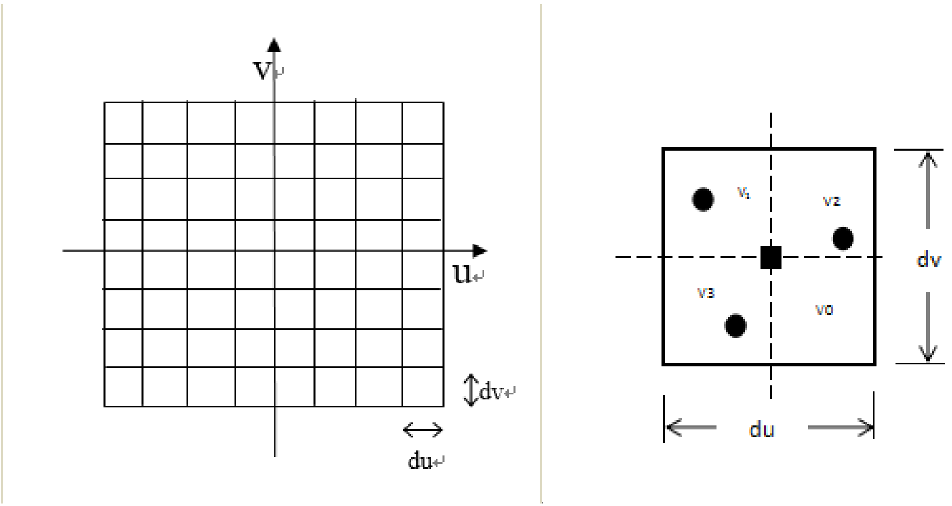



图 1 均匀划分的网格（左）与非均匀分布的可见度数据示意图（右）
Fig.1 Regular grid (left) and the schematic diagram of non-uniform visibility data (right)

将非均匀采样可见度数据转换成均匀采样可见度数据，即对未知像素点的数据进行重采样，主要有两类方法：内插法和卷积核法。一些内插值方法被开发用于重建非均匀采样，如最邻近插值，双线性插值等；文献[15]表明最优的网格化方法是卷积一个无限扩展的 sinc 函数，即卷积核法，出于实际考虑，有限的卷积函数取代了无限扩展的 sinc 函数。常用的网格卷积函数（Gridding Convolution Function，GCF）有截断 sinc 函数，截断指数函数，拟合的球面波函数，文献[16]讨论了这些网格卷积函数在不同尺寸大小下的性能。在 MUSER 数据处理中，我们选取了大小为 6 的拟合的球面波函数作为网格卷积函数。

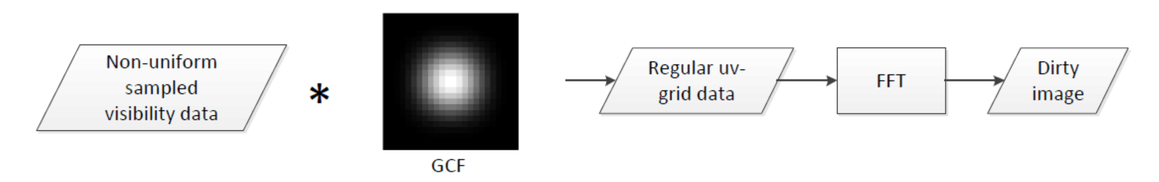



图 2 卷积网格化并成像
Fig.2 Convolutional gridding and imaging

## 1.2 Gridding 算法并行化

根据 OpenCL 编程规范，OpenCL 程序主要由两部分组成：一部分是采用 OpenCL 语言（类 C 语言）编写的内核函数，执行在 OpenCL 设备（多核 CPU、GPU、Cell、DSP、FPGA 等）上；另一部分是通过调用 OpenCL 的应用编程接口（Application Programing Interface，API）管理内核函数的主机程序，执行在主机（CPU）上。

本文将 gridding 过程中的关键步骤，由 OpenCL 语言编写成内核函数，以便在 OpenCL 设备

上并行执行，再通过在主机上调用这些内核函数，从而实现 gridding 算法的并行化。主要改写成的内核函数有：gridding_kernel，correct_grid_kernel，normalize_kernel，point_symmetric_kernel，shift_kernel 等。此外，成像过程的傅里叶变换是通过调用基于 OpenCL 实现的快速傅里叶变换 FFT，包含于 python 的 pyfft 包中。

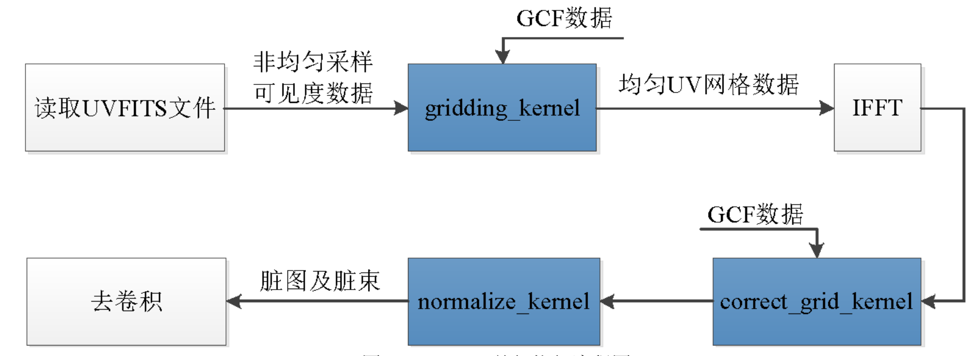



图 3 Gridding 并行执行流程图

Fig. 3 The flow diagram of parallel gridding

本文处理的~~图像~~数据是以二维数组的形式表示，在采用 OpenCL 语言编写内核函数过程中，由 OpenCL 提供的二维数组索引 get_global_id(0)和 get_global_id(1)，唯一标识二维数组中的元素，为方便计算，需借助传递给内核函数的二维数组的参数（宽度或高度），将二维数组按行或按列转换成一维数组，这样，二维索引 ~~get_global_id(0)和 get_global_id(1)~~转换成一维索引~~值~~ id，当内核函数在并行设备中执行时，每个线程处理由索引~~值~~ id 唯一标识的二维数组元素。

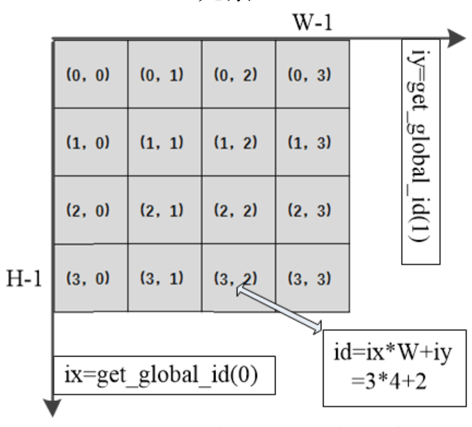



图 4 OpenCL 二维索引转换一维索引示意图

Fig. 4 The schematic diagram of OpenCL two-dimensional index to one-dimensional index

在 gridding 执行过程中，由于可见度数据共轭对称，故在对可见度数据进行插值~~网格化~~时，为减少~~网格化的~~计算量，先对一半的可见度数据进行插值~~网格化~~，再通过中心点对称方式计算得到另一半数据。将中心点对称操作编写成内核函数 point_symmetric_kernel，示意图如下。

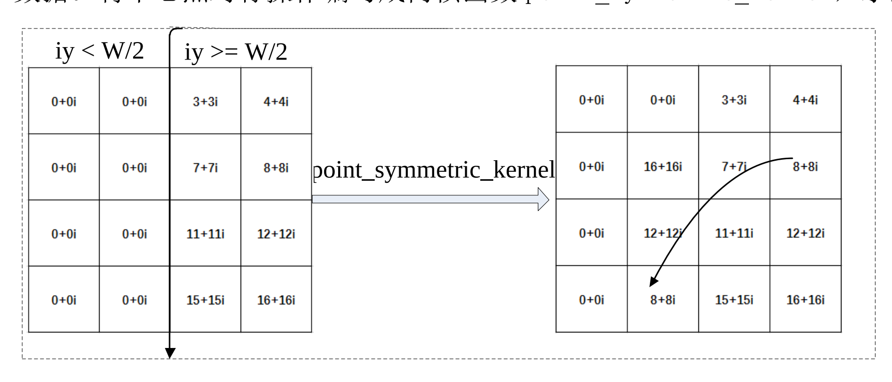


图 5 中心点对称操作示意图

Fig.5 The schematic diagram of point symmetric operation

point_symmetric_kernel 伪代码如下：

```
OpenCL kernel function: point_symmetric_kernel
Input parameter: a two dimensional array im, a flag hfac, the two dimensional array height H and width W
//Obtain two dimensional array index (ix,iy)
ix←get_global_id(0)
iy←get_global_id(1)
// Stay within the bounds
If ix > 0 and ix < H and iy > 0 and iy < W/2 then
    nix←H-ix
    niy←W-iy
    im[ix*W + iy].x←im[nix*W + niy].x
    im[ix*W + iy].y←hfac*im[nix*W + niy].y
Output: a new two dimensional array im
```

在傅里叶变换过程中，为使变换后的图像是一个完整的周期，需要进行平移（shift）操作。将 shift 操作编写成内核函数 shift_kernel，示意图如下。

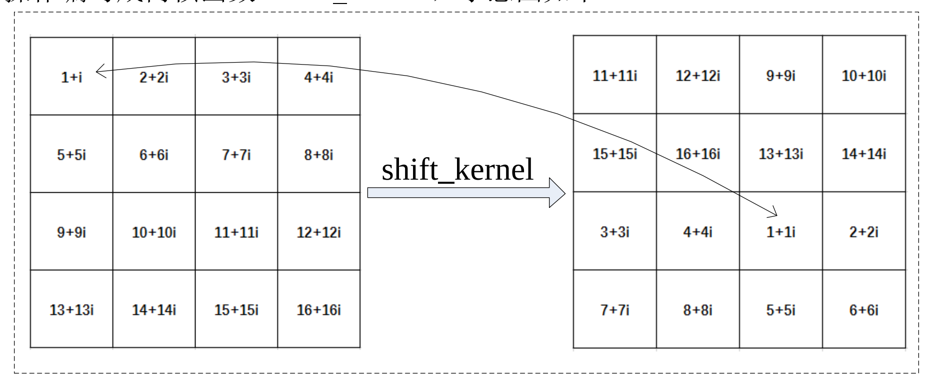



图 6 傅里叶变换平移操作示意图

Fig.6 The schematic diagram of Fourier transform shift operation

在 gridding 执行过程中，由于可见度数据与一个网格卷积函数 GCF 进行了卷积，为消除 GCF 的影响，需进行网格校正操作。将网格校正操作编写成内核函数 correct_grid_kernel。~~此外，~~进行网格校正后，需要对脏图和脏束~~进一步的~~进行归一化或标准化。~~将脏束进行归一化，~~即将脏图和脏束中所有元素除以脏束中的最大值~~，对脏图进行标准化，即以及将脏图中所有所以元素除以脏束中的最大值，~~从而保证脏束的峰值为 1 以及脏图与脏束具有相同量纲~~标准化后的脏束和脏图具有相同的量纲~~，便于后续进行洁化。

| | | | |
|---|---|---|---|
| 1 | 4 | 5 | 1 |
| 1 | 3 | 4 | 1 |
| 1 | 2 | 3 | 1 |
| 2 | 1 | 2 | 1 |

normalize_kernel

| | | | |
|---|---|---|---|
| 1*num | 4*num | 5*num | 1*num |
| 1*num | 3*num | 4*num | 1*num |
| 1*num | 2*num | 3*num | 1*num |
| 2*num | 1*num | 2*num | 1*num |

~~图 7 标准化操作示意图~~

~~Fig.7 The schematic diagram of normalize operation~~

## 1.3 并行加权

由于 UV 采样点太少，对可见度函数采样值进行傅里叶逆变换得到的图像（~~称为~~脏图）包含一些虚假信息，通过对 UV 采样点赋予不同的权值来改善图像质量的操作称之为加权[17]。常用的加权方式有自然加权，统一加权以及 Taper ~~加权~~，本文将这三种加权方式改写成 OpenCL 内核函数 weight_natural_kernel、weight_uniform_kernel 和 weight_taper_kernel。

三种加权方法伪代码如下：

```
OpenCL kernel function: weight_natural_kernel、weight_uniform_kernel、weight_taper_kernel
Input parameter: a two dimensional array nim, the number of samples falling into cell cnt, the two dimensional array height H and width W
//Obtain two dimensional array index (ix,iy)
ix←get_global_id(0)
iy←get_global_id(1)
id←iy + ix*W
// Stay within the bounds
If ix > 0 and ix < H and iy > 0 and iy < W then
    if cnt[id] != 0 then
         //natural weighting
         wgt←1.
         //uniform weighting
         wgt←1./ cnt[id]
         //taper weighting
         wgt←exp(-((ix-H/2)*(ix-H/2)+(iy-W/2)*(iy-W/2))/(2*0.5*0.5*W*W))
        nim[id].x←nim[id].x*wgt
        nim[id].y←nim[id].y*wgt
Output: weighted nim
```

自然加权赋予相应像素点的权重为1，统一加权赋予相应像素点的权重为区域内采样点频数的倒数，Taper加权赋予相应像素点的权重为对应高斯函数值。通过改变像素点的权重大小，对可见度数据进行加权操作，从而改善图像质量。

## 2 实验结果

### 2.1 实验环境

OpenCL 定义不同的公司或厂商为不同的平台(Platform)，每个公司或厂商提供不同的 OpenCL 设备(Device)。本文选用的 OpenCL 设备有 NVIDIA 的 GPU (GeForce GTX TITAN X，Tesla k20m)和 Intel 的 CPU (Intel Xeon E5-2620 V2)，平台及设备参数见表 2。

**表 2 平台和设备参数信息**

**Tab.2 The information of platform and device**

| Parameter | Intel CPU | GTX TITAN X GPU | Tesla k20m GPU |
|---|---|---|---|
| Platform vendor | Intel(R) Corporation | NVIDIA Corporation | NVIDIA Corporation |
| Platform version | OpenCL 1.2 LINUX | OpenCL 1.2 CUDA 7.5.18 | OpenCL 1.2 CUDA 7.5.18 |
| Device name | Intel(R) Xeon(R) CPU E5-2620 V2 @ 2.10GHz | GeForce GTX TITAN X | Tesla k20m |
| Device type | CPU | GPU | GPU |
| Device max clock speed | 2100 MHz | 1215 MHz | 705 MHz |
| Device compute units | 24 | 24 | 13 |
| Device cores | 6 | 2496 | 2496 |
| Device max work group size | 8192 | 1024 | 1024 |

### 2.2 实验结果

实验数据来源于 MUSER 2015 年 11 月 1 日 12 时 8 分 49 秒 354 毫秒时刻对太阳的观测数据，原始观测数据经过一系列预处理（坏天线标记，星表查询等）合成为标准天文数据格式(UVFITS 文件)，数据文件被命名为 20151101-120849_354161240.uvfits。通过 MUSER 数据处理系统执行网格化和快速傅里叶变换等操作，生成相应时刻太阳亮度分布图像（未经过去卷积操作的图像称为脏图）以及对应的点扩散函数（Point Spread Function，PSF)，PSF 也称为脏束。

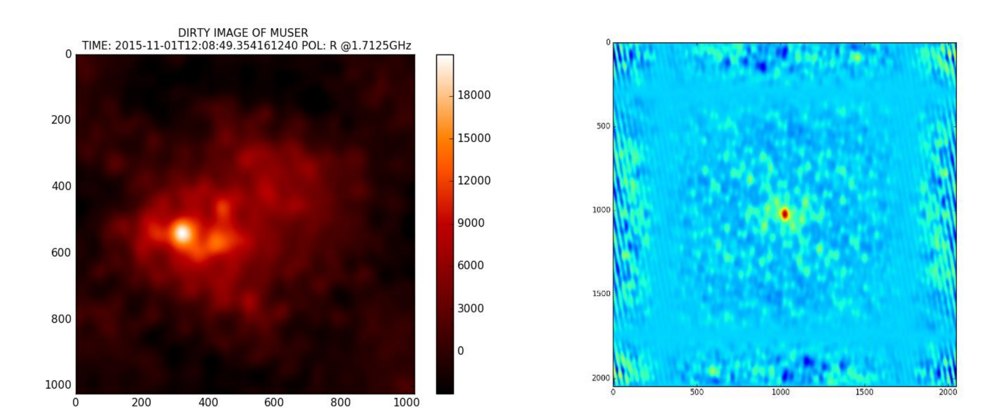



图 ~~8~~7 MUSER 成的脏图（左）和点扩散函数 PSF（右），观测时间是 2015 年 11 月 1 日 12:08:49:354，频率是 1.7125 GHz，极化方式是右旋

Fig. ~~8~~7 The dirty image (left) and PSF (right) of MUSER at 12:08:49:354 on November 11th, 2015 (Frequency: 1.7125 GHz, Polarization: right)

在同一台服务器上，分别在 CPU（Intel Xeon E5-2620 V2）、GPU（GeForce GTX TITAN X, Tesla k20m）并行设备上执行基于 OpenCL 实现的 gridding 算法和在 GPU（GeForce GTX TITAN X, Tesla k20m）上执行基于 CUDA 实现的 gridding 算法，并进行快速傅里叶变换成图，分别生成大小为 1024*1024 pixels 和 2048*2048 pixels 的脏图。网格化并成图（Gridding+IFFT）过程执行时间参照下表。

**表 3 Gridding+IFFT 过程执行时间表**

**Tab.3 the Execution time of Gridding and FFT**

| ~~Device~~ 设备 | ~~Image size~~ 图像大小 | ~~Execution time (s)~~ 执行时间（s） |
|---|---|---|
| CPU OpenCL | 1024*1024 | 0.904 |
| | 2048*2048 | 4.407 |
| GPU OpenCL(TITAN X) | 1024*1024 | 0.350 |
| | 2048*2048 | 0.713 |
| GPU OpenCL(k20m) | 1024*1024 | 0.372 |
| | 2048*2048 | 0.782 |
| GPU CUDA(TITAN X) | 1024*1024 | 0.323 |
| | 2048*2048 | 0.610 |
| GPU CUDA(k20m) | 1024*1024 | 0.344 |
| | 2048*2048 | 0.760 |

从表中可以看出，执行基于 OpenCL 和基于 CUDA 实现的 gridding 算法并成图，在 GPU 环境下所消耗的时间基本相当，同时，基于 OpenCL 实现的 gridding 算法还能在纯 CPU 环境下较快速地被执行。

## 2.3 讨论

本文采用并行计算 OpenCL 技术，基于 Python 语言导入 PyOpenCL 与 PyFFT 扩展包，对 MUSER 成像过程中的 gridding 算法以及快速傅里叶变换 FFT 成图过程进行了实现。基于 OpenCL 实现的 gridding 算法具有如下优点：

（1）能够在 GPU 环境中执行，并且不局限与 NVIDIA GPU（实验条件限制，只选用了 NVIDIA GPU 和 Intel CPU），执行效率与基于 CUDA 实现的 gridding 算法执行效率大致相当，保证了 MUSER 成图的性能；

（2）能够在多核 CPU 环境下执行，适用于在单机环境下进行开发与测试，便于 MUSER 软件的推广与应用。

综上所述，采用 OpenCL 实现的并行 gridding 算法，在保证算法执行效率的同时有效地提升了算法对硬件平台的适应性。此外，本工作进一步验证了 OpenCL 在天文软件开发中的可用性，从 CUDA 到 OpenCL，异构系统从 NVIDIA GPU+CPU 的异构模式转变成并行设备+CPU 的异构模式。这种异构模式的转变将扩展并行计算在天文高性能软件开发中的应用。

## 3 下一步工作

本文研究的射电干涉阵成像过程局限于小视场，在大视场成像中 w 项引起的视场扭曲是不可忽略的，传统的二维 FFT 成像将不再适用，因此，下一步我们将在此研究工作的基础上，采用 OpenCL 技术对大视场成像中的关键算法，例如 w-projection，faceting 以及大视场混合算法，进行并行优化。

## 4 致谢

衷心感谢以下项目对本项工作的资助。基金项目：国家重点研发计划(2016YFE0100300)，国家自然科学基金委员会-中国科学院天文联合基金(No.U1531132,U1631129,U1231205)，国家自然科学基金(No.11403009,11463003,11773012)，赛尔网络下一代互联网技术创新项目(No.NGII20170204)。

感谢国家天文台-阿里云天文大数据联合研究中心对本项工作的支持。



## 参考文献

[1] Yan Y, Zhang J, Wang W, et al. The Chinese Spectral Radioheliograph—CSRH[J]. Earth Moon & Planets, 2009, 104(1-4):97-100.
[2] 赖铖，梅盈，邓辉,等. MUSER 可见度数据积分方法与实现[J]. 天文研究与技术, 2018(1):78-86.
Lai Cheng, Mei Ying, Deng Hui, et al. Integral Method and Implementation of MUSER Visibility Data[J]. Astronomical Research & Technology——Publications of National Astronomical Observatories of China, 2018(1):78-86.
[3] 周鑫磊，王威，王锋,等. 基于QT的MUSER观测数据多屏图形化实时显示的设计与实现[J]. 天文研究与技术, 2015, 12(4):503-509.
Zhou Xinlei, Wang Wei, Wang Feng, et al. Design and Implementation of a Multi-Monitor Display System Based on the QT for NAOC MUSER Observations [J]. Astronomical Research & Technology——Publications of National Astronomical Observatories of China, 2015, 12(4): 503-509.
[4] Shi C, Wang F, Deng H, et al. High Performance Negative Database for Massive Data Management System of The Mingantu Spectral Radioheliograph[J]. Publications of the Astronomical Society of the Pacific, 2017, 129(978).
[5] Dai H M, Mei Y, Wang W, et al. An Auto-flag Method of Radio Visibility Data Based on Support Vector Machine [J]. Chinese Astronomy & Astrophysics, 2017, 41(1):125-135.
[6] Feng W, Hui D, Wei W. High performance distributed data processing pipeline for Chinese Spectral RadioHeliograph[C]// Radio Science Conference. IEEE, 2015:1-1.
[7] Wei S, Wang F, Deng H, et al. OpenCluster: A Flexible Distributed Computing Framework for Astronomical Data Processing[J]. Publications of the Astronomical Society of the Pacific, 2016, 129(972):024001.
[8] Wang F, Ji K F. Distributed Data-Processing Pipeline for Mingantu Ultrawide Spectral Radioheliograph[J]. Publications of the Astronomical Society of the Pacific, 2015, 127(950):383-396.
[9] 陈泰燃，王威，王锋,等. 基于MPI的高性能UVFITS数据合成研究与应用[J]. 天文研究与技术, 2016, 13(2):184-189.
Chen Tairan, Wang Wei, Wang Feng, et al. The Study and Application of a High Performance UVFITS Assembly System Based on MPI[J].Astronomical Research & Technology——Publications of National Astronomical Observatories of China, 2016, 13(2):184-189.
[10] Mei Y, Wang F, Wang W, et al. GPU-Based High-Performance Imaging for Mingantu Spectral RadioHeliograph[J]. 2017.
[11] 冯勇，陈坤，邓辉,等. 基于OpenCL的MUSERCLEAN算法研究与实现[J]. 天文学报, 2017, 58(2):55-64.
Feng Yong, Chen Kun, Deng Hui, et al. The Research and Implementation of MUSER CLEAN Algorithm Based on OpenCL[J]. Acta Astronomica Sinica, 2017, 58.
[12] Högbom J A. Aperture Synthesis with a Non-Regular Distribution of Interferometer Baselines[J]. Astronomy & Astrophysics Supplement, 2011, 15(15):417.
[13] Thompson A R, Bracewell R N. Interpolation and Fourier transformation of fringe visibilities[J]. Astronomical Journal, 1973, 79(1):11-24.
[14] Sedarat H, Nishimura D G. On the optimality of the gridding reconstruction algorithm[J]. IEEE Transactions on Medical Imaging, 2000, 19(4):306.
[15] O'Sullivan J D. A Fast Sinc Function Gridding Algorithm for Fourier Inversion in Computer Tomography[J]. IEEE Transactions on Medical Imaging, 1985, 4(4):200-207.
[16] Jackson J I, Meyer C H, Nishimura D G, et al. Selection of a Convolution Function for Fourier Inversion Using Gridding[C]// IEEE Trans. Medical Imaging. 1991:473-478.
[17] Boone F. Weighting interferometric data for direct imaging[J]. Experimental Astronomy, 2013, 36(1-2):77-104.

# Implementation of gridding algorithm for radio interferometric imaging based on OpenCL

Feng Yong[1], Wang Feng[1,2,3], Deng Hui[1,2], Wei Shoulin[1], Mei Ying[1,2,3], Dai Wei[1,3], Shi CongMing[1]

(1. Computer Technology Application Key Lab of Yunnan Province, Kunming University of Science and Technology, Kunming 650500, China

2. Astrophysics Center/Institute of Physics and Electronic Engineering, Guangzhou University Guangzhou, Guangzhou 510006, China

3. Yunnan Astronomical Observatories, Chinese Academy of Sciences, Kunming 650011, China)

**Abstract:** It's urgent to carry out high-performance scientific data processing with a single machine in the development and application of astronomical software. However, due to the different configurations of machines, the traditional CUDA + GPU technology has obvious limitations in portability and seamlessness. According to gridding algorithm in MingantU SpEctral Radioheliograph (MUSER) data processing, the OpenCL technology is used in parallel to implement multi-thread programming. The experimental results show that the gridding algorithm based on OpenCL can not only can run on ~~various~~ GPUs, but also merely on CPUs. While choosing execution on GPU, the execution efficiency of gridding algorithm is approximately equal with it based on CUDA. At the same time, the algorithm is not limited to the NVIDIA's GPU, which has solved the problem of environmental dependence of CUDA+GPU. And the algorithm also has an acceptable execution efficiency ~~high implementation~~ with the merely CPU, which is suitable for development and testing astronomy software with a single machine ~~, but also~~ and will facilitate the application and promotion of astronomical software.

**Key words:** Gridding; Parallel computing; OpenCL; MUSER